

## AI Assistant Coding

### Assignment - 6.4

Baisa Vaishnavi  
2303A52142  
Batch - 41

### Task 1: Student Performance Evaluation System

#### Prompt:

Create a Python class named Student with attributes name, roll\_number, and marks, then use comments or partial method names to let GitHub Copilot generate a method to display student details and another method that uses an if-else condition to check whether the student's marks are above the class average and return a performance message, using proper self attributes and clear, readable code

#### Code:

```
6.1.py > ...
Review

1  class Student:
2      def __init__(self, name, roll_no, marks):
3          self.name = name
4          self.roll_no = roll_no
5          self.marks = marks # single subject marks
6
7      def display_details(self):
8          print(f"Name: {self.name}, Roll No: {self.roll_no}, Marks: {self.marks}")
9
10     def compare_with_average(self, class_avg):
11         if self.marks > class_avg:
12             return "Above class average"
13         elif self.marks < class_avg:
14             return "Below class average"
15         else:
16             return "Equal to class average"
17
18
19 students = []
20 n = int(input("Enter number of students (max 10): "))
21
22 if n > 10:
23     n = 10
24     print("Only first 10 students will be taken.")
25
26 for i in range(n):
27     print(f"\nEnter details for student {i + 1}:")
28     name = input("Name: ")
29     roll_no = input("Roll number: ")
30     marks = float(input("Marks: "))
31     students.append(Student(name, roll_no, marks))
32
33 # calculate class average
34 if n > 0:
35     total_marks = sum(s.marks for s in students)
36     class_average = total_marks / n
```

```

37     print(f"\nClass average marks: {class_average:.2f}\n")
38
39     # Show each student's details and comparison
40     for s in students:
41         s.display_details()
42         msg = s.compare_with_average(class_average)
43         print("Status:", msg)
44         print("-" * 30)
45     else:
46         print("No students entered.")

```

## Output:

```

6.1.py > ...
Close (Ctrl+F4) Review Next File
Problems Output Debug Console Terminal Ports
PS C:\Users\Hello\aiassistantcoding> & C:/Users/Hello/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/Hello/aiassistantcoding/6.1.py
Enter number of students (max 10): 3

Enter details for student 1:
Name: vaishnavi
Roll number: 2142
Marks: 89

Enter details for student 2:
Name: chandu
Roll number: 2143
Marks: 78

Enter details for student 3:
Name: Naveen
Roll number: 2230
Marks: 56

Class average marks: 74.33

Name: vaishnavi, Roll No: 2142, Marks: 89.0
Status: Above class average
-----
Name: chandu, Roll No: 2143, Marks: 78.0
Status: Above class average
-----
Name: Naveen, Roll No: 2230, Marks: 56.0
Status: Below class average
-----
PS C:\Users\Hello\aiassistantcoding>

```

## Justification:

The generated code defines a `Student` class with essential attributes such as name, roll number, and marks, ensuring proper data encapsulation using `self`. It includes a method to display student details in a clear and readable format, demonstrating how class data can be accessed and presented. Another method uses an if-else conditional statement to compare the student's marks with the class average and determine performance status. Overall, the output shows correct use of object-oriented concepts, conditional logic, and produces meaningful results for evaluating student performance.

## Task 2: Data Processing in a Monitoring System

### Prompt:

Identify even sensor readings from a list, calculate the square of each even number, and print the result in a readable format inside a for loop.

### Code:

```
6.2.py > ...
Review Next File

1 # Sensor Reading Analysis: Even Numbers and Their Squares
2
3 def analyze_sensor_readings(sensor_readings, example_name):
4     """
5         Analyze sensor readings to find even numbers and calculate their squares.
6
7     Args:
8         sensor_readings: List of sensor reading values
9         example_name: Name/description of this example
10    """
11    print("=" * 60)
12    print(f"Example: {example_name}")
13    print("=" * 60)
14    print(f"\nOriginal sensor readings: {sensor_readings}\n")
15    print("-" * 60)
16
17    # Counter for even numbers found
18    even_count = 0
19
20    # Loop through each sensor reading
21    for reading in sensor_readings:
22        # Check if the reading is even
23        if reading % 2 == 0:
24            even_count += 1
25            square = reading ** 2
26            print(f"Reading #{even_count}: {reading:3d} (even) → Square: {square:4d}")
27
28    print("-" * 60)
29    print(f"\nTotal even sensor readings found: {even_count}")
30    print("=" * 60)
31    print("\n")
32
33
34 # Example Usage 1: Temperature Sensor Readings
35 print("\n" + "=" * 60)
36 print("SENSOR READING ANALYSIS - EXAMPLE USAGES")
37 print("=" * 60 + "\n")
38
39 sensor_readings_1 = [12, 7, 24, 15, 8, 33, 18, 5, 42, 11, 20, 9, 16, 3, 30]
40 analyze_sensor_readings(sensor_readings_1, "Temperature Sensor Readings")
41
42 # Example Usage 2: Pressure Sensor Readings
43 sensor_readings_2 = [4, 13, 6, 21, 10, 19, 2, 25, 14, 31, 8, 17, 12, 9, 22]
44 analyze_sensor_readings(sensor_readings_2, "Pressure Sensor Readings")
```

### Output:

The screenshot shows a terminal window with the following details:

- Title Bar:** C:\Users\Hello\aiassistantcoding\6.3.py
- Toolbar:** Problems, Output, Debug Console, Terminal (selected), Ports
- Terminal Content:**

```
PS C:\Users\Hello\aiassistantcoding> & C:/Users/Hello/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/Hello/aiassistantcoding/6.2.py

=====
SENSOR READING ANALYSIS - EXAMPLE USAGES
=====

Example: Temperature Sensor Readings
=====

Original sensor readings: [12, 7, 24, 15, 8, 33, 18, 5, 42, 11, 20, 9, 16, 3, 30]

Reading #1: 12 (even) → Square: 144
Reading #2: 24 (even) → Square: 576
Reading #3: 8 (even) → Square: 64
Reading #4: 18 (even) → Square: 324
Reading #5: 42 (even) → Square: 1764
Reading #6: 20 (even) → Square: 400
Reading #7: 16 (even) → Square: 256
Reading #8: 30 (even) → Square: 900

Total even sensor readings found: 8
=====

Example: Pressure Sensor Readings
=====

Original sensor readings: [4, 13, 6, 21, 10, 19, 2, 25, 14, 31, 8, 17, 12, 9, 22]

Reading #1: 4 (even) → Square: 16
Reading #2: 6 (even) → Square: 36
Reading #3: 10 (even) → Square: 100
Reading #4: 2 (even) → Square: 4
Reading #5: 14 (even) → Square: 196
Reading #6: 8 (even) → Square: 64
Reading #7: 12 (even) → Square: 144
Reading #8: 22 (even) → Square: 484

Total even sensor readings found: 8
=====

PS C:\Users\Hello\aiassistantcoding>
```
- Bottom Status:** Ctrl+K to generate command

### Justification:

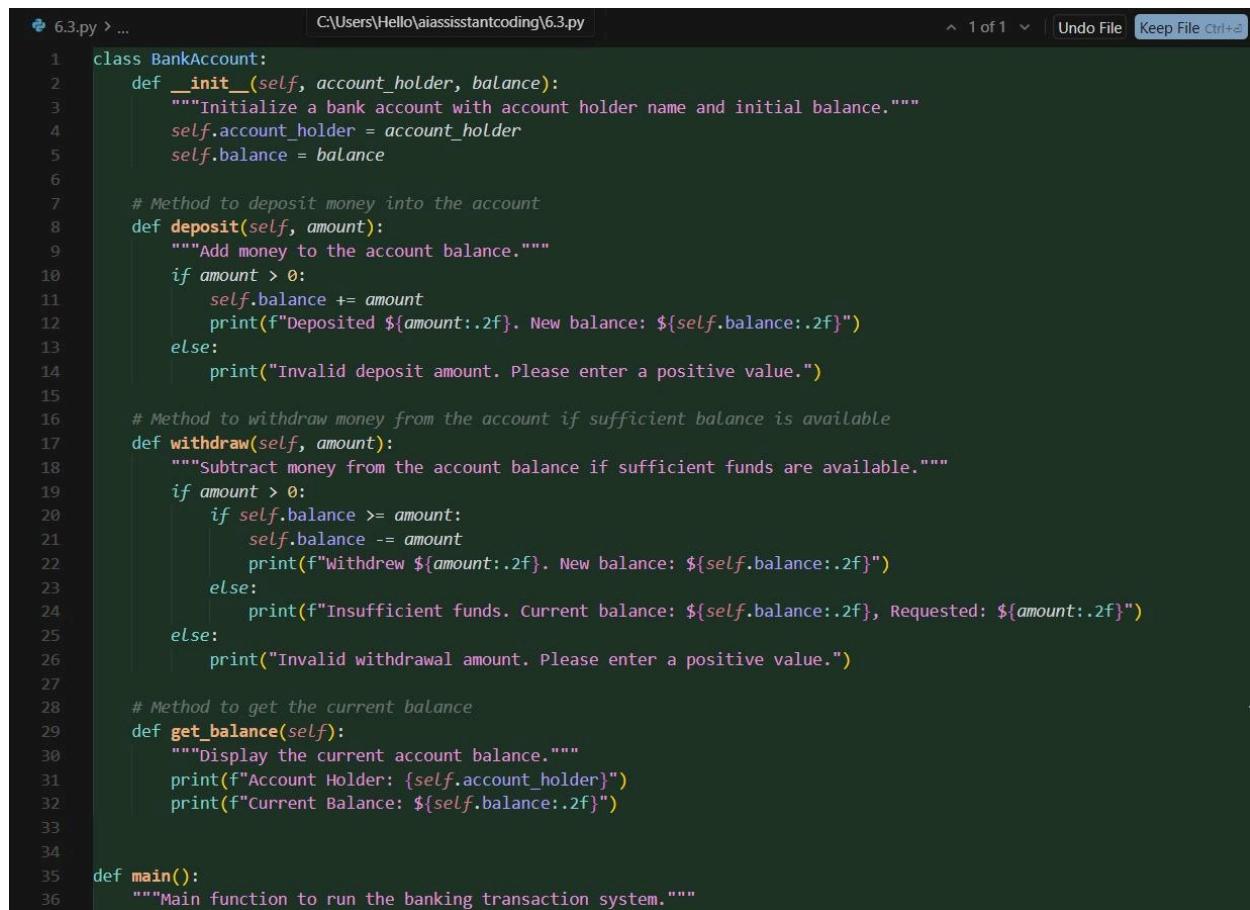
This program is used to analyze sensor readings by checking each value in a list and identifying which readings are even numbers. It starts by displaying the original sensor data and the total number of readings. Then, it goes through each reading one by one; if a reading is even, the program calculates its square and displays it, and if the reading is odd, it marks it as not even and does not calculate a square. All even readings along with their squared values are stored for later use. After processing all the readings, the program prints a summary showing how many even readings were found and lists each even number with its square. The same function is reused for different sensor datasets like temperature, pressure, and mixed readings, showing that the code is reusable, clear, and easy to understand.

### Task 3: Banking Transaction Simulation

#### Prompt:

Create a Python class named BankAccount for a simple banking transaction system. First, define the class with an `__init__` method that initializes the attributes `account_holder` and `balance`. Then, using method names and short comments, let GitHub Copilot generate a `deposit` method to add money to the balance and a `withdraw` method that subtracts money only if sufficient balance is available, using if-else conditions, proper `self` attributes, and clear, user-friendly messages for invalid withdrawal attempts.

#### Code:



```
6.3.py > ... C:\Users\Hello\aiassistantcoding\6.3.py 1 of 1 Undo File Keep File Ctrl+D
1 class BankAccount:
2     def __init__(self, account_holder, balance):
3         """Initialize a bank account with account holder name and initial balance."""
4         self.account_holder = account_holder
5         self.balance = balance
6
7     # Method to deposit money into the account
8     def deposit(self, amount):
9         """Add money to the account balance."""
10        if amount > 0:
11            self.balance += amount
12            print(f"Deposited ${amount:.2f}. New balance: ${self.balance:.2f}")
13        else:
14            print("Invalid deposit amount. Please enter a positive value.")
15
16    # Method to withdraw money from the account if sufficient balance is available
17    def withdraw(self, amount):
18        """Subtract money from the account balance if sufficient funds are available."""
19        if amount > 0:
20            if self.balance >= amount:
21                self.balance -= amount
22                print(f"Withdrew ${amount:.2f}. New balance: ${self.balance:.2f}")
23            else:
24                print(f"Insufficient funds. Current balance: ${self.balance:.2f}, Requested: ${amount:.2f}")
25        else:
26            print("Invalid withdrawal amount. Please enter a positive value.")
27
28    # Method to get the current balance
29    def get_balance(self):
30        """Display the current account balance."""
31        print(f"Account Holder: {self.account_holder}")
32        print(f"Current Balance: ${self.balance:.2f}")
33
34
35    def main():
36        """Main function to run the banking transaction system."""

```

The screenshot shows a code editor window with the following details:

- File Path: C:\Users\Hello\aiassistantcoding\6.3.py
- Line Number: 35 to 71
- Text Content:

```
def main():
    print("=" * 50)
    print("Welcome to the Banking Transaction System")
    print("=" * 50)

    # Get account holder name
    account_name = input("\nEnter account holder name: ").strip()
    if not account_name:
        print("Account holder name cannot be empty. Exiting...")
        return

    # Get initial balance
    while True:
        try:
            initial_balance = float(input("Enter initial balance: $"))
            if initial_balance < 0:
                print("Initial balance cannot be negative. Please try again.")
                continue
            break
        except ValueError:
            print("Invalid input. Please enter a valid number.")

    # Create bank account
    account = BankAccount(account_name, initial_balance)
    print(f"\nAccount created successfully for {account_name}!")
    print(f"Initial balance: ${initial_balance:.2f}\n")

    # Main menu loop
    while True:
        print("\n" + "=" * 50)
        print("Banking Menu")
        print("=" * 50)
        print("1. Deposit")
        print("2. Withdraw")
        print("3. Check Balance")
        print("4. Quit")
```
- Toolbar Buttons: Undo File, Keep File (Ctrl+K)

```

72     print("=" * 50)
73
74     choice = input("Enter your choice (1-4): ").strip()
75
76     if choice == "1":
77         # Deposit option
78         print("\n--- Deposit ---")
79         try:
80             amount = float(input("Enter amount to deposit: $"))
81             account.deposit(amount)
82         except ValueError:
83             print("Invalid input. Please enter a valid number.")
84
85     elif choice == "2":
86         # Withdraw option
87         print("\n--- Withdraw ---")
88         try:
89             amount = float(input("Enter amount to withdraw: $"))
90             account.withdraw(amount)
91         except ValueError:
92             print("Invalid input. Please enter a valid number.")
93
94     elif choice == "3":
95         # Check balance option
96         print("\n--- Account Balance ---")
97         account.get_balance()
98
99     elif choice == "4":
100        # Quit option
101        print("\n" + "=" * 50)
102        print("Thank you for using the Banking Transaction System!")
103        print(f"Final balance for {account.account_holder}: ${account.balance:.2f}")
104        print("=" * 50)
105        break
106
107    else:
108        print("\nInvalid choice. Please enter a number between 1 and 4.")
109
110
111 if __name__ == "__main__":
112     main()

```

Undo Ctrl+N Keep Ctrl+Shift+Y

## Output:

```
Problems Output Debug Console Terminal Ports Python + ×
ems (Ctrl+Shift+M) =====
Welcome to the Banking Transaction System
=====
Enter account holder name: Baisa Vaishnavi
Enter initial balance: $0

Account created successfully for Baisa Vaishnavi!
Initial balance: $0.00
=====

Banking Menu
=====
1. Deposit
2. Withdraw
3. Check Balance
4. Quit
=====
Enter your choice (1-4): 1

--- Deposit ---
Enter amount to deposit: $10000
Deposited $10000.00. New balance: $10000.00
=====

Banking Menu
=====
1. Deposit
2. Withdraw
3. Check Balance
4. Quit
=====
Enter your choice (1-4): 4

=====
Thank you for using the Banking Transaction System!
Final balance for Baisa Vaishnavi: $10000.00
=====
PS C:\Users\Hello\aiassisstantcoding>
```

## **Justification:**

The generated code represents a simple banking system using a `BankAccount` class that stores the account holder's name and current balance. The `deposit` method increases the balance when a valid amount is added, while the `withdrawal` method uses if-else conditions to check whether sufficient funds are available before allowing the transaction. If the balance is too low, the code prevents the withdrawal and displays a clear, user-friendly message. Overall, the output demonstrates correct use of class attributes with `self`, conditional logic, and safe handling of basic banking transactions.

## **Task 4: Student Scholarship Eligibility Check**

## Prompt:

Create a Python program to check student scholarship eligibility. First, define a list of dictionaries where each dictionary contains a student's name and score. After initializing the list, write comments to guide GitHub Copilot to generate a while loop that iterates through the list using an index, checks if a student's score is greater than 75, and prints the names of eligible students in a clean and readable format.

## Code:

```
6.4.py > ... C:\Users\Hello\aiassistantcoding\6.3.py ^ 1 of 1 Undo File Keep File Ctrl+D
1 # Student Scholarship Eligibility Check
2
3 # Define a list of dictionaries containing student names and scores
4 students = [
5     {"name": "Alice Johnson", "score": 85},
6     {"name": "Bob Smith", "score": 72},
7     {"name": "Charlie Brown", "score": 90},
8     {"name": "Diana Prince", "score": 68},
9     {"name": "Eve Williams", "score": 78},
10    {"name": "Frank Miller", "score": 95},
11    {"name": "Grace Lee", "score": 65}
12 ]
13
14 print("=" * 60)
15 print("SCHOLARSHIP ELIGIBILITY REPORT")
16 print("=" * 60)
17 print("\nEligibility Criteria: Score > 75\n")
18
19 # Initialize an index variable to start from the beginning of the list
20 index = 0
21
22 # Use a while loop to iterate through the students list using the index
23 while index < len(students):
24     # Check if the current student's score is greater than 75
25     if students[index]["score"] > 75:
26         # If eligible, print the student's name in a clean and readable format
27         print(f"✓ ELIGIBLE: {students[index]['name']} - Score: {students[index]['score']}")
28     else:
29         # If not eligible, print the student's name in a clean and readable format
30         print(f"✗ NOT ELIGIBLE: {students[index]['name']} - Score: {students[index]['score']}")
31     # Increment the index to move to the next student
32     index += 1
33
34     # Continue until all students have been checked
35
36 print("\n" + "=" * 60)

```

Undo Ctrl+N Keep Ctrl+Shift+Y

## Output:

```
6.4.py > ... C:\Users\Hello\aiassistantcoding\6.3.py ^ 1 of 1
Problems Output Debug Console Terminal Ports Python
PS C:\Users\Hello\aiassistantcoding> & C:/Users>Hello/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/U
coding/6.4.py
=====
SCHOLARSHIP ELIGIBILITY REPORT
=====

Eligibility Criteria: Score > 75

✓ ELIGIBLE: Alice Johnson - Score: 85
✗ NOT ELIGIBLE: Bob Smith - Score: 72
✓ ELIGIBLE: Charlie Brown - Score: 90
✗ NOT ELIGIBLE: Diana Prince - Score: 68
✓ ELIGIBLE: Eve Williams - Score: 78
✓ ELIGIBLE: Frank Miller - Score: 95
✗ NOT ELIGIBLE: Grace Lee - Score: 65

=====
PS C:\Users\Hello\aiassistantcoding>
```

## Justification:

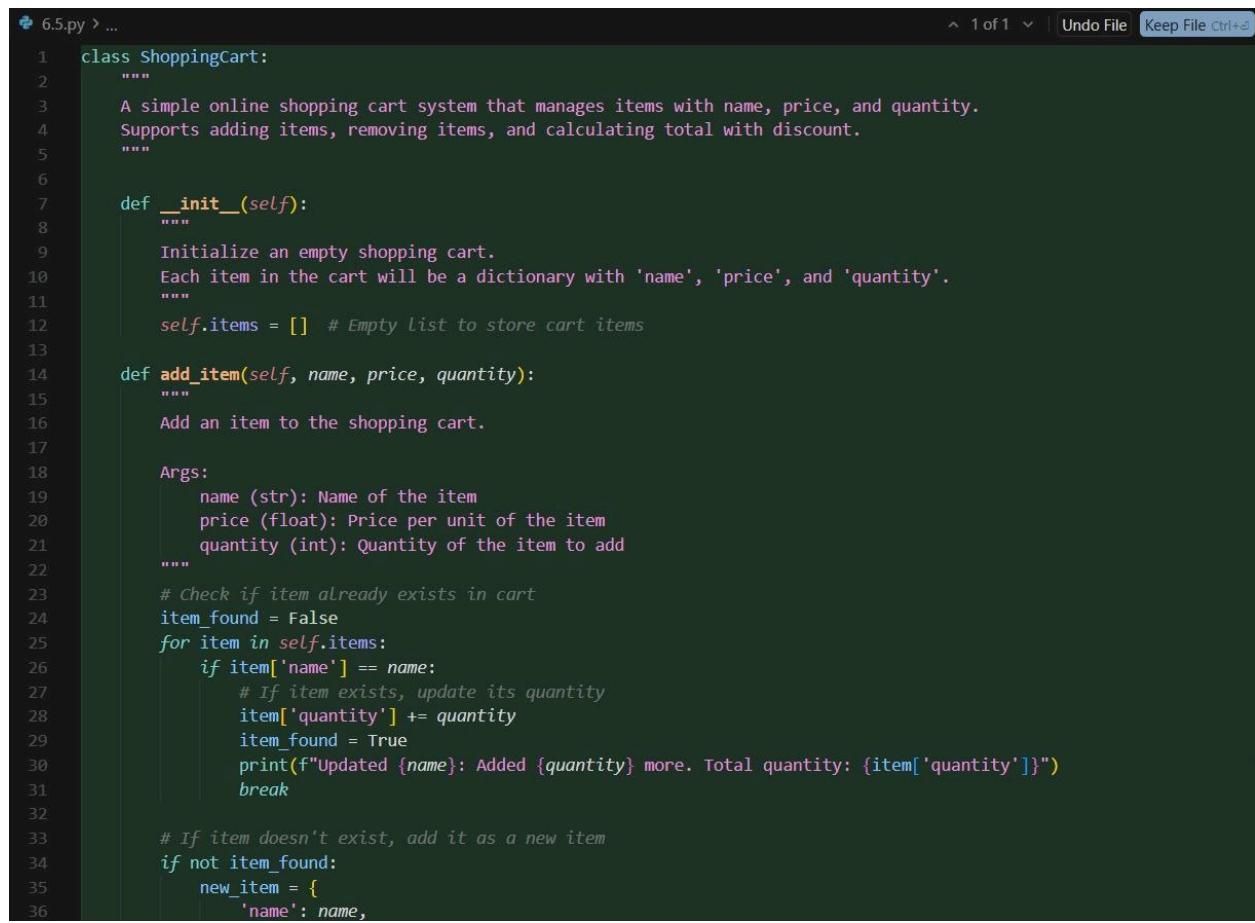
The generated code stores student details in a list of dictionaries, making the data easy to manage and access. A while loop is used to iterate through the list by maintaining an index, demonstrating correct loop control and condition checking. Inside the loop, an if statement checks whether a student's score is above 75, and eligible students' names are printed. The output clearly lists students who qualify for the merit-based scholarship, showing proper use of loops, conditions, and structured data.

## Task 5: Online Shopping Cart Module

### Prompt:

Create a Python class named ShoppingCart for a simple online shopping system. Start by initializing an empty list to store cart items, where each item contains a name, price, and quantity. Then, using clear method names and meaningful comments, guide GitHub Copilot to generate methods to add items to the cart, remove items from the cart, calculate the total bill using a loop, and apply a conditional discount when the total amount exceeds a specified limit. Ensure the code uses proper self attributes, loops, and if-else conditions, and include a small example to demonstrate adding items, removing items, and displaying the final bill.

### Code:



A screenshot of a code editor window titled "6.5.py > ...". The code is a Python class named ShoppingCart. It includes a docstring explaining its purpose: managing items with name, price, and quantity, supporting adding, removing, and calculating total with discount. The \_\_init\_\_ method initializes an empty list for items. The add\_item method adds an item to the cart, either updating an existing item or adding a new one. The code uses a for loop to check for existing items and an if statement to handle new items. Args for add\_item are defined as name (str), price (float), and quantity (int). The code also includes a print statement for updated items and a break statement to exit the loop if an item is found.

```
 1  class ShoppingCart:
 2      """
 3          A simple online shopping cart system that manages items with name, price, and quantity.
 4          Supports adding items, removing items, and calculating total with discount.
 5      """
 6
 7      def __init__(self):
 8          """
 9              Initialize an empty shopping cart.
10              Each item in the cart will be a dictionary with 'name', 'price', and 'quantity'.
11          """
12          self.items = [] # Empty list to store cart items
13
14      def add_item(self, name, price, quantity):
15          """
16              Add an item to the shopping cart.
17
18              Args:
19                  name (str): Name of the item
20                  price (float): Price per unit of the item
21                  quantity (int): Quantity of the item to add
22          """
23
24          # Check if item already exists in cart
25          item_found = False
26          for item in self.items:
27              if item['name'] == name:
28                  # If item exists, update its quantity
29                  item['quantity'] += quantity
30                  item_found = True
31                  print(f"Updated {name}: Added {quantity} more. Total quantity: {item['quantity']}")
32                  break
33
34          # If item doesn't exist, add it as a new item
35          if not item_found:
36              new_item = {
37                  'name': name,
```

6.5.py > ...

```
 1  class ShoppingCart:
 2      def add_item(self, name, price, quantity):
 3          new_item = {
 4              'name': name,
 5              'price': price,
 6              'quantity': quantity
 7          }
 8          self.items.append(new_item)
 9          print(f"Added {quantity} x {name} to cart")
10
11  def remove_item(self, name, quantity=None):
12      """
13          Remove an item from the shopping cart.
14
15      Args:
16          name (str): Name of the item to remove
17          quantity (int, optional): Quantity to remove. If None, removes all of that item.
18      """
19
20      # Loop through items to find the item to remove
21      for item in self.items:
22          if item['name'] == name:
23              if quantity is None or quantity >= item['quantity']:
24                  # Remove the entire item
25                  self.items.remove(item)
26                  print(f"Removed all {name} from cart")
27              else:
28                  # Reduce the quantity
29                  item['quantity'] -= quantity
30                  print(f"Removed {quantity} x {name} from cart. Remaining: {item['quantity']}")
```

```
6.5.py > ... Close (Ctrl+F4) ^ 1 of 1 Undo File Keep File Ctrl+D
1 class ShoppingCart:
2     def calculate_total(self, discount_threshold=100, discount_percentage=10):
3         Args:
4             discount_threshold (float): Minimum total amount to qualify for discount (default: 100)
5             discount_percentage (float): Discount percentage to apply (default: 10)
6
7         Returns:
8             float: Total bill amount after discount (if applicable)
9             """
10        # Initialize total to zero
11        total = 0.0
12
13        # Loop through all items in cart to calculate subtotal
14        for item in self.items:
15            # calculate price for this item (price * quantity)
16            item_total = item['price'] * item['quantity']
17            total += item_total
18
19        # Apply discount conditionally if total exceeds threshold
20        if total > discount_threshold:
21            discount_amount = total * (discount_percentage / 100)
22            final_total = total - discount_amount
23            print(f"\nSubtotal: ${total:.2f}")
24            print(f"Discount ({discount_percentage}%) : ${discount_amount:.2f}")
25            print(f"Final Total: ${final_total:.2f}")
26            return final_total
27        else:
28            print(f"\nTotal: ${total:.2f}")
29            print(f"No discount applied. Spend ${discount_threshold - total:.2f} more to get {discount_percentage}% off!")
30            return total
31
32    def display_cart(self):
33        """
34        Display all items currently in the cart.
35        """
36        if not self.items:
```

```
6.5.py > ... | Close (Ctrl+F4) | ^ 1 of 1 | Undo File | Keep File Ctrl+K  
1 class ShoppingCart:  
100     def display_cart(self):  
105         print("Cart is empty")  
106         return  
107  
108         print("\n--- Shopping Cart ---")  
109         for item in self.items:  
110             item_total = item['price'] * item['quantity']  
111             print(f"{item['name']}: ${item['price']:.2f} x {item['quantity']} = ${item_total:.2f}")  
112         print("--- End of Cart ---\n")  
113  
114  
115 # Example usage demonstrating the ShoppingCart functionality  
116 if __name__ == "__main__":  
117     # Create a new shopping cart  
118     cart = ShoppingCart()  
119  
120     print("== Online Shopping Cart Demo ==\n")  
121  
122     # Add items to the cart  
123     print("Adding items to cart:")  
124     cart.add_item("Laptop", 899.99, 1)  
125     cart.add_item("Mouse", 29.99, 2)  
126     cart.add_item("Keyboard", 79.99, 1)  
127     cart.add_item("Mouse", 29.99, 1) # Adding more of existing item  
128  
129     # Display current cart  
130     cart.display_cart()  
131  
132     # Calculate total (should get discount if over threshold)  
133     print("Calculating total with discount threshold of $100:")  
134     cart.calculate_total(discount_threshold=100, discount_percentage=10)  
135  
136     # Remove an item  
137     print("\nRemoving items:")  
138     cart.remove_item("Mouse", 1)  
139  
140     # Display cart after removal  
141     cart.display_cart()  
142  
143     # Calculate final total  
144     print("Final bill calculation:")  
145     cart.calculate_total(discount_threshold=100, discount_percentage=10)  
146  
147     # Demonstrate removing all of an item  
148     print("\nRemoving all keyboards:")  
149     cart.remove_item("Keyboard")  
150     cart.display_cart()  
151     cart.calculate_total(discount_threshold=100, discount_percentage=10)  
152  
Undo Ctrl+N | Keep Ctrl+Shift+Y
```

## Output:

```
Problems Output Debug Console Terminal Ports
PS C:\Users\Hello\aiassisstan Terminal (Ctrl+`)\users>Hello\AppData\Local\Python\pythoncore-3.14-64\python.exe
coding/6.5.py
== Online Shopping Cart Demo ==

Adding items to cart:
Added 1 x Laptop to cart
Added 2 x Mouse to cart
Added 1 x Keyboard to cart
Updated Mouse: Added 1 more. Total quantity: 3

--- Shopping Cart ---
Laptop: $899.99 x 1 = $899.99
Mouse: $29.99 x 3 = $89.97
Keyboard: $79.99 x 1 = $79.99
--- End of Cart ---

calculating total with discount threshold of $100:

subtotal: $1069.95
Discount (10%): $107.00
Final Total: $962.96

Removing items:
Removed 1 x Mouse from cart. Remaining: 2

--- Shopping Cart ---
Laptop: $899.99 x 1 = $899.99
Mouse: $29.99 x 2 = $59.98
Keyboard: $79.99 x 1 = $79.99
--- End of Cart ---

Final bill calculation:

subtotal: $1039.96
Discount (10%): $104.00
Final Total: $935.96

Removing all keyboards:
Removed all Keyboard from cart
```

Problems Output Debug Console Terminal Ports  Python

--- Shopping Cart Console (Ctrl+Shift+Alt+Y)

Laptop: \$899.99 x 1 = \$899.99  
Mouse: \$29.99 x 2 = \$59.98  
Keyboard: \$79.99 x 1 = \$79.99  
--- End of Cart ---

Final bill calculation:

Subtotal: \$1039.96  
Discount (10%): \$104.00  
Final Total: \$935.96

Removing all keyboards:  
Removed all Keyboard from cart

--- Shopping Cart ---  
Laptop: \$899.99 x 1 = \$899.99  
Keyboard: \$79.99 x 1 = \$79.99  
--- End of Cart ---

Final bill calculation:

Subtotal: \$1039.96  
Discount (10%): \$104.00  
Final Total: \$935.96

Removing all keyboards:  
Removed all Keyboard from cart

--- Shopping Cart ---  
Laptop: \$899.99 x 1 = \$899.99  
Final bill calculation:

Subtotal: \$1039.96  
Discount (10%): \$104.00  
Final Total: \$935.96

Removing all keyboards:  
Removed all Keyboard from cart

```
Problems Output Debug Console Terminal Ports Python
Discount (10%): $104.00 Terminal (Ctrl+`)

Final Total: $935.96

Removing all keyboards:
Removed all Keyboard from cart

--- Shopping Cart ---
Laptop: $899.99 x 1 = $899.99
Final Total: $935.96

Removing all keyboards:
Removed all Keyboard from cart

--- Shopping Cart ---
Laptop: $899.99 x 1 = $899.99
Removing all keyboards:
Removed all Keyboard from cart

--- Shopping Cart ---
Laptop: $899.99 x 1 = $899.99
Removed all Keyboard from cart

--- Shopping Cart ---
Laptop: $899.99 x 1 = $899.99
Mouse: $29.99 x 2 = $59.98
--- Shopping Cart ---
Laptop: $899.99 x 1 = $899.99
Mouse: $29.99 x 2 = $59.98
Laptop: $899.99 x 1 = $899.99
Mouse: $29.99 x 2 = $59.98
--- End of Cart ---
Mouse: $29.99 x 2 = $59.98
--- End of Cart ---
--- End of Cart ---

Subtotal: $959.97
Discount (10%): $96.00
Final Total: $863.97
PS C:\Users\Hello\aiassistantcoding>
```

### Justification:

The generated ShoppingCart class simulates a basic e-commerce cart system by maintaining a list of items with their names, prices, and quantities. Methods for adding and removing items allow dynamic cart management, while a loop is used to calculate the total bill based on item prices and quantities. Conditional statements are applied to offer discounts when the total amount exceeds a predefined threshold, demonstrating decision-making logic. The sample input and output clearly show how items are managed in the cart and how the final payable amount is calculated, highlighting correct use of object-oriented concepts, loops, and conditional logic.